Calcolo Scientifico

a.a. 2007-2008

Installazione e utilizzo della libreria BLAS

Dove scaricare BLAS?

http://www.netlib.org

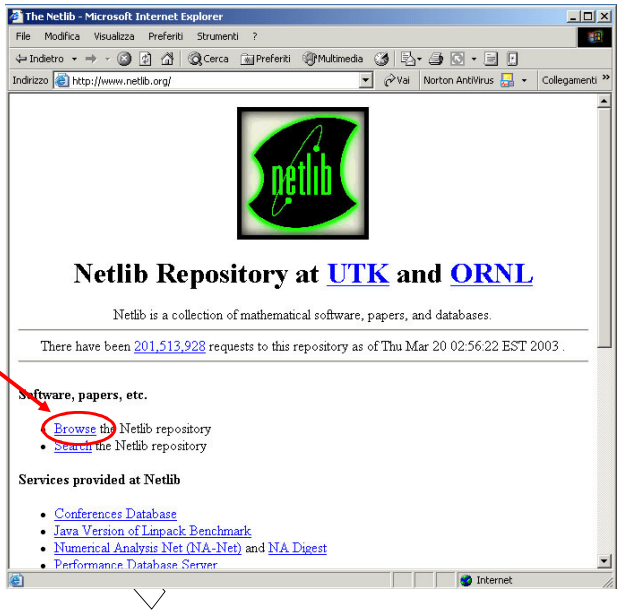
È un repository, una collezione di software matematico, papers e database

Realizzato nel 1985 da Jack Dongarra e Eric Grosse, AT&T Bell Labs

Motivato dall'esigenza di avere a disposizione software matematico

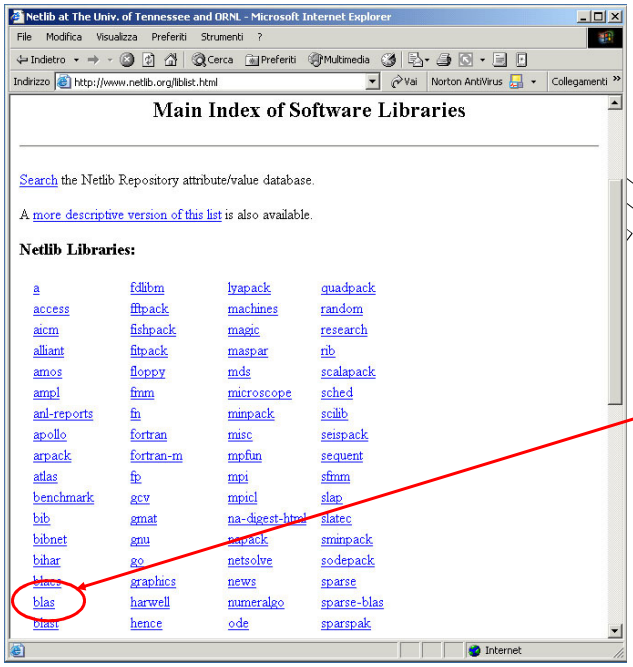
* libero (free)
* di alta qualità
* in breve tempo

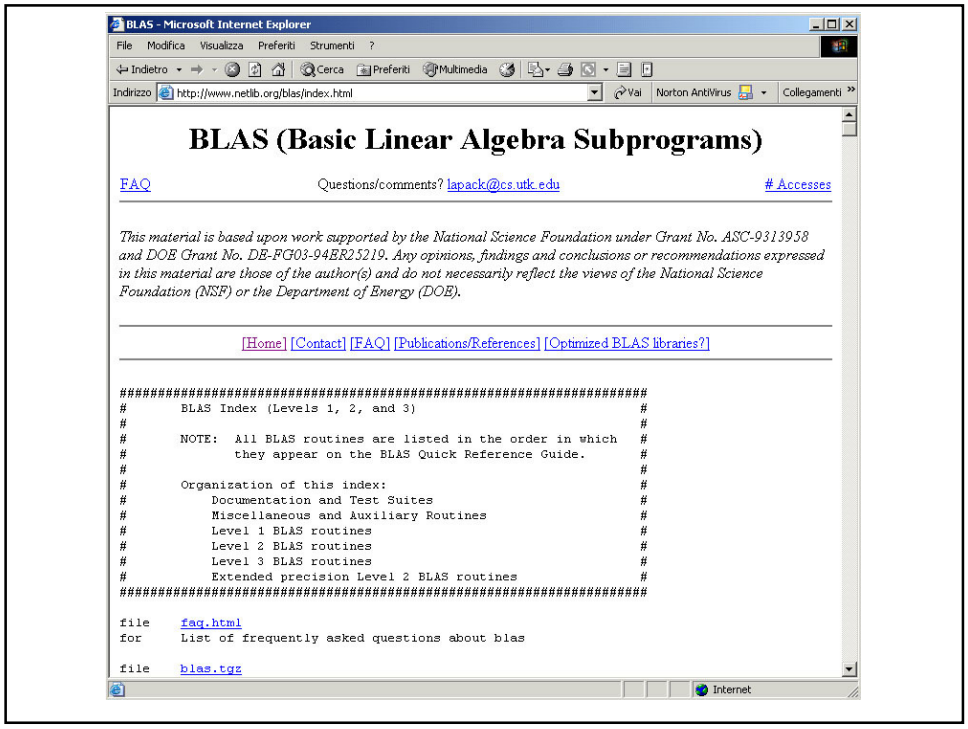
per accedere a BLAS

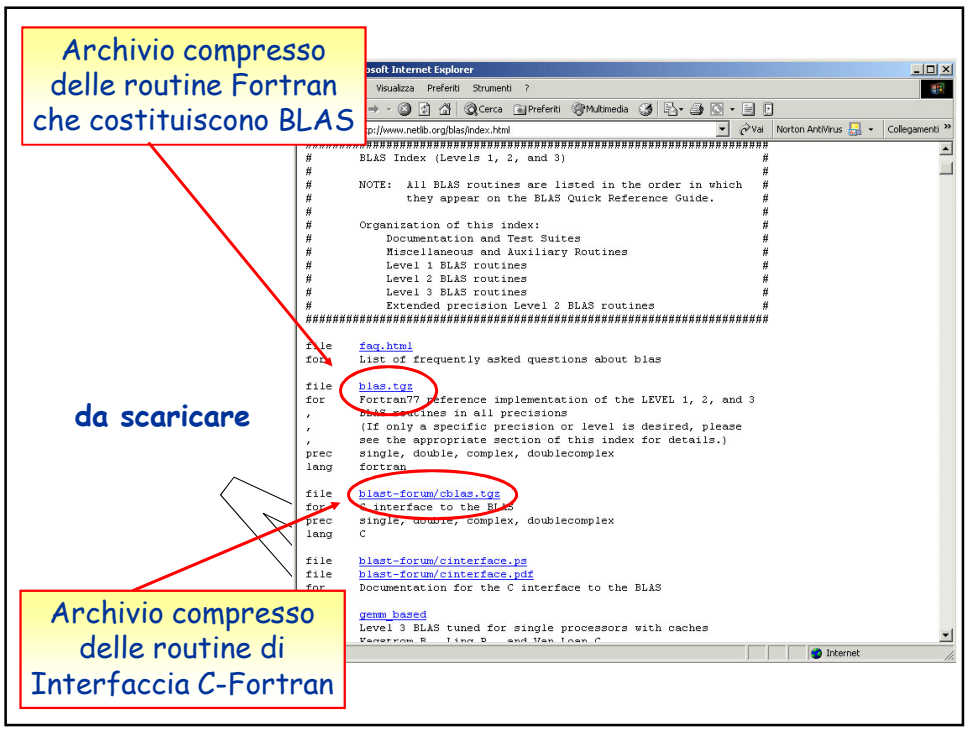


Compare, in ordine alfbetico, l'elenco del sofware disponibile

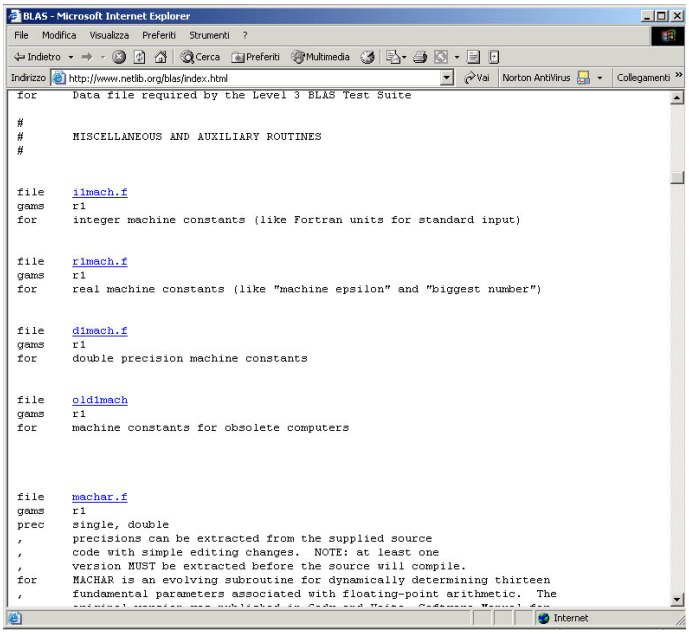
BLAS




BLAS - Microsoft Internet Explorer
http://www.netlib.org/blas/index.html
BLAS (Basic Linear Algebra Subprograms)
FAQ
Questions/comments? lapack@cs.utk.edu
Accesses
This material is based upon work supported by the National Science Foundation under Grant No. ASC-9313958 and DOE Grant No. DE-FG03-94ER25219. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF) or the Department of Energy (DOE).
[Home] [Contact] [FAQ] [Publications/References] [Optimized BLAS libraries?]
BLAS Index (Levels 1, 2, and 3)
NOTE: All BLAS routines are listed in the order in which they appear on the BLAS Quick Reference Guide.
Organization of this index:
Documentation and Test Suites
Miscellaneous and Auxiliary Routines
Level 1 BLAS routines
Level 2 BLAS routines
Level 3 BLAS routines
Extended precision Level 2 BLAS routines
file faq.html
for List of frequently asked questions about blas
file blas.tgz
Internet


Archivio compresso delle routine Fortran che costituiscono BLAS
da scaricare
Archivio compresso delle routine di Interfaccia C-Fortran
file faq.html
file blas.tgz
file blast-forum/cblas.tgz
file blast-forum/cinterface.ps
file blast-forum/cinterface.pdf
Documentation for the C interface to the BLAS
gemm_based
Level 3 BLAS tuned for single processors with caches

Inoltre devono essere scaricate alcune routine ausiliarie.



Installare BLAS

S.O. Linux

1. Creare una libreria contenente le routines compilate di BLAS.
   a. Decomprimere il file blas.tgz

      `prompt> gunzip blas.tgz` ⇒ blas.tar

   b. Scompattare file blas.tar

      `prompt> tar -xvf blas.tar` ⇒ file.f

   c. Compilare i file.f

      `prompt> g77 -c *.f` ⇒ file.o

   d. Creare la libreria libblas.a

      `prompt> ar -q libblas.a *.o` ⇒ libblas.a

Come utilizzare BLAS

È possibile da un programma scritto in C richiamare un programma scritto in Fortran?

SI

ma con qualche accorgimento

In Fortran
il passaggio di variabili
avviene per indirizzo

In C
il passaggio di variabili
avviene per valore
o per indirizzo

bisogna passare le variabili per indirizzo

Un programma: copia di un vettore in un altro

```
#include <stdio.h>
void dcopy(int *,double ,int *, double ,int *);

main( )
{
 int n,i,incx,incy;
 double *dx,*dy;

 incx=1;
 incy=1;

 printf("inserire la dimensione\n");
 scanf("%d",&n);

 dx=(double *)calloc(n,sizeof(double));
 dy=(double *)calloc(n,sizeof(double));

 for(i=0;i<n;i++)
  dx[i]=i;

 dcopy_(&n,dx,&incx,dy,&incy);

 for(i=0;i<n;i++)
  printf("%lf\n",dy[i]);

 free(dx);
 free(dy);
}
```

per compilare:

```
prompt> gcc –o ese prova.c –Lpath -lblas
```

dove per path si intende il path della directory dove si trova la libreria *libblas.a*

Installare CBLAS

2. Installare l'interfaccia C-Fortran
   a. Decomprimere il file cblas.tgz

```
prompt> gunzip cblas.tgz
```

→ cblas.tar

   b. Scompattare file cblas.tar

```
prompt> tar -xvf cblas.tar
```

→ DIR CBLAS

   c. Copiare la libreria blas creata

```
prompt> cp libblas.a CBLAS/lib/
```

   d. Creare il makefile per l'architettura che si intende utilizzare

```
prompt> cd CBLAS
prompt> ln -s Makefile.LINUX  Makefile.in
```

   c. Compilare

```
prompt> make alllib
```

Il Makefile

```
#
# Makefile.LINUX
#
#
# If you compile, change the name to Makefile.in.
#

#--------------------------------------------------------------------
# Shell
#--------------------------------------------------------------------
SHELL = /bin/sh
#--------------------------------------------------------------------
# Platform
#--------------------------------------------------------------------
PLAT = LINUX
#--------------------------------------------------------------------
# Libraries and includs
#--------------------------------------------------------------------
BLLIB = $(HOME)/CBLAS/lib/libblas.a
CBDIR = $(HOME)/CBLAS
CBLIBDIR = $(CBDIR)/lib/$(PLAT)
CBLIB = $(CBLIBDIR)/cblas_$(PLAT).a
#--------------------------------------------------------------------
# Compilers
#--------------------------------------------------------------------
CC = gcc
FC = g77
LOADER = $(FC)
#--------------------------------------------------------------------
# Flags for Compilers
#--------------------------------------------------------------------
CFLAGS = -O3 -DADD_
FFLAGS = -O3
#--------------------------------------------------------------------
# Archive programs and flags
#--------------------------------------------------------------------
ARCH = ar
ARCHFLAGS = r
RANLIB = echo
```

Lo schema dei nomi

Al nome della routine Fortran bisogna aggiungere il prefisso cblas_

Così, ad esempio, la routine *saxpy* diventa *cblas_saxpy*

Un programma

y=x

```
#include <stdio.h>
#include "cblas.h"

main( )
{
 int n,i,incx,incy;
 double *dx,*dy;

 incx=1;
 incy=1;

 printf("inserire la dimensione\n");
 scanf("%d",&n);

 dx=(double *)calloc(n,sizeof(double));
 dy=(double *)calloc(n,sizeof(double));

 for(i=0;i<n;i++)
  dx[i]=i;

 cblas_dcopy(n,dx,incx,dy,incy);

 for(i=0;i<n;i++)
  printf("%lf\n",dy[i]);

 free(dx);
 free(dy);
}
```

Alcune notazioni

Tutti gli argomenti che sono caratteri nell'interfaccia Fortran sono trattati in C come variabili intere.

Gli argomenti carattere in Fortran sono:
SIDE, UPLO, TRANSPOSE, DIAG.

L'interfaccia utilizza una ulteriore variabile per gestire gli array bidimensionali: *ORDER.*

```
enum CBLAS_ORDER     {CblasRowMajor=101, CblasColMajor=102};
enum CBLAS_TRANSPOSE {CblasNoTrans=111, CblasTrans=112, CblasConjTrans=113};
enum CBLAS_UPLO      {CblasUpper=121, CblasLower=122};
enum CBLAS_DIAG      {CblasNonUnit=131, CblasUnit=132};
enum CBLAS_SIDE      {CblasLeft=141, CblasRight=142};
```

Gli array bidimensionali

Nell'interfaccia C non vengono utilizzati array bidimensionali.
Una matrice di dimensione m×n viene memorizzata, per righe o per colonne, in un array monodimensionale di dimensione m*n

Se la matrice è memorizzata per righe
Order=CblasRowMajor=101

Se la matrice è memorizzata per colonne
Order=CblasColMajor=102

Se il parametro Order è posto uguale CblasRowMajor, si assume che gli elementi appartenenti ad una stessa riga sono contigui in memoria, mentre gli elementi di una stessa colonna sono separati in memoria da un numero costante di elementi.
Tale parametro corrisponde alla *leading dimension* (LDA) in Fortran.

La leading dimension

In Fortran le matrici, array bidimensionali, vengono memorizzate per colonne.

In Fortran l'allocazione della memoria è statica.

In fase di dichiarazione viene allocato un array bidimensionale di dimensione uguale o superiore a quella che verrà realmente utilizzata.

Supponiamo di dover utilizzare una matrice di dimensione massima 4.

```
...
real a(4,4)
...
```

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

in memoria

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}	a_{13}	a_{23}	a_{33}	a_{43}	a_{14}	a_{24}	a_{34}	a_{44}

Supponiamo di far girare il programma con una matrice di dimensione n=2

```
...
   integer i,j,n
   real a(4,4)
   n=2
   do 20 i=1,n
    do 10 j=1,n
      read *,a(i,j)
10  continue
20 continue
...
```

1	2		
3	4		

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}	a_{13}	a_{23}	a_{33}	a_{43}	a_{14}	a_{24}	a_{34}	a_{44}
1	3			2	4										

lda

Quando chiamiamo un'altra procedura viene passato solo l'indirizzo del primo elemento dell'array a, a_{11}. È quindi necessario conoscere la dimensione delle colonne (il numero di righe con cui a è stata dichiarata= **lda**)

Esempio

```
...
   integer i,j,n
   real a(4,4)
   n=2
   do 20 i=1,n
    do 10 j=1,n
      read *,a(i,j)
10  continue
20 continue
...
call opmat(a,n,...)
...
```

```
subroutine opmat (a,n...)
integer i,j,n
real a(n,n),x
...
 x=3*a(1,2)
...
```

In generale, l'istruzione a(i,j) fa riferimento alla locazione di memoria a(1+(i-1)+(j-1)*2)

i=1, j=2 noi intendiamo

1	2		
3	4		

viene letto

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}	a_{13}	a_{23}	a_{33}	a_{43}	a_{14}	a_{24}	a_{34}	a_{44}
1	3			2	4										

Esempio

```
...
   integer i,j,n,lda
   real a(4,4)
   lda=4
   n=2
   do 20 i=1,n
    do 10 j=1,n
      read *,a(i,j)
10  continue
20 continue
...
call opmat(a,n,lda...)
...
```

```
Subroutine opmat (a,n,lda...)
integer i,j,n,lda
real a(lda,n),x
...
 x=3*a(1,2)
...
```

L'istruzione a(i,j) fa riferimento alla locazione di memoria a(i+(j-1)*lda)

i=1, j=2 noi intendiamo

1	2		
3	4		

viene letto

a_{11}	a_{21}	a_{31}	a_{41}	a_{12}	a_{22}	a_{32}	a_{42}	a_{13}	a_{23}	a_{33}	a_{43}	a_{14}	a_{24}	a_{34}	a_{44}
1	3			2	4										

In definitiva

È necessario specificare la distanza tra due successivi elementi appartenenti alla stessa riga.

lda

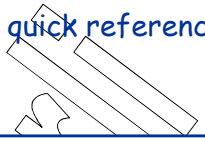
leading dimension

y=Ax

```
#include <stdio.h>
#include "cblas.h"
main()
{
 int m,n,i,j,incx,incy,lda,or,tra;
 double *A,*x,*y,alfa,beta;

 incx=1;
 incy=1;

 printf("inserire il numero di righe della matrice A\n");
 scanf("%d",&m);
 printf("inserire il numero di colonne della matrice A\n");
 scanf("%d",&n);
 lda=n;
 alfa=1.;
 beta=0.;
 or=101;   /* matrice memorizzata per righe */
 tra=111;  /* no trasposta */

 A=(double *)calloc(m*n,sizeof(double));
 x=(double *)calloc(n,sizeof(double));
 y=(double *)calloc(n,sizeof(double));

 for(i=0;i<m;i++)
   for(j=0;j<n;j++)
     scanf("%lf",&A[i*n+j]);

 for(i=0;i<n;i++)
   scanf("%lf",x[i]);

 cblas_dgemv(or,tra,m,n,alfa,A,lda,x,incx,beta,y,incy);

 for(i=0;i<n;i++)
  printf("%lf\n",y[i]);

 free(A);
 free(x);
 free(y);
}
```

Come ricordare tutti i nomi delle routine?

Utilizzando le quick reference



Level 1 BLAS

	dim	scalar	vector	vector	scalars	5-element array		prefixes
SUBROUTINE xROTG (					A, B, C, S)		Generate plane rotation	S, D
SUBROUTINE xROTMG(					D1, D2, A, B,	PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (	N,		X, INCX,	Y, INCY,	C, S)		Apply plane rotation	S, D
SUBROUTINE xROTM (	N,		X, INCX,	Y, INCY,		PARAM)	Apply modified plane rotation	S, D
SUBROUTINE xSWAP (	N,		X, INCX,	Y, INCY)			$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (	N,	ALPHA,	X, INCX)				$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (	N,		X, INCX,	Y, INCY)			$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (	N,	ALPHA,	X, INCX,	Y, INCY)			$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (	N,		X, INCX,	Y, INCY)			$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (	N,		X, INCX,	Y, INCY)			$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (	N,		X, INCX,	Y, INCY)			$dot \leftarrow x^H y$	C, Z
FUNCTION xxDOT (	N,		X, INCX,	Y, INCY)			$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (	N,		X, INCX)				$nrm2 \leftarrow \|\|x\|\|_2$	S, D, SC, DZ
FUNCTION xASUM (	N,		X, INCX)				$asum \leftarrow \|\|re(x)\|\|_1 + \|\|im(x)\|\|_1$	S, D, SC, DZ
FUNCTION IxAMAX(	N,		X, INCX)				$amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|im(x_k)\| = max(\|re(x_i)\| + \|im(x_i)\|)$	S, D, C, Z

Level 2 BLAS

	options			dim	b-width	scalar	matrix	vector	scalar	vector		
xGEMV (		TRANS,		M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xGBMV (		TRANS,		M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xHEMV (	UPLO,			N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHBMV (	UPLO,			N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHPMV (	UPLO,			N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xSYMV (	UPLO,			N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSBMV (	UPLO,			N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSPMV (	UPLO,			N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xTRMV (	UPLO,	TRANS,	DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTBMV (	UPLO,	TRANS,	DIAG,	N, K,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTPMV (	UPLO,	TRANS,	DIAG,	N,			AP,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTRSV (	UPLO,	TRANS,	DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTBSV (	UPLO,	TRANS,	DIAG,	N, K,			A, LDA,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z
xTPSV (	UPLO,	TRANS,	DIAG,	N,			AP,	X, INCX)			$x \leftarrow A^{-1}x, x \leftarrow A^{-T}x, x \leftarrow A^{-H}x$	S, D, C, Z